

Project One: Perceptron Demo

Bradley Reeves
Sam Shissler



Department of Computer Science
Central Washington University
April 22, 2021

Contents

1	Introduction	3
2	Datasets	4
3	Experimentation	6
4	Analysis	7
5	Conclusion	9

List of Figures

1.1	McCulloch and Pitts' Neuron.	3
1.2	The Perceptron network.	4
2.1	I input samples (3×3 matrix).	5
2.2	L input samples (3×3 matrix).	5
2.3	I input samples (5×5 matrix).	6
2.4	L input samples (5×5 matrix).	6
4.1	Hyper-parameters with the best accuracy.	7

List of Tables

2.1	Dataset Generation Rules.	5
4.1	Effect of learning rate on training.	8

Introduction

This project aims to explore the Perceptron neural network model. The Perceptron is a single-layer network based on the McCulloch and Pitts' mathematical model of a neuron. This model is shown in Figure 1.1.

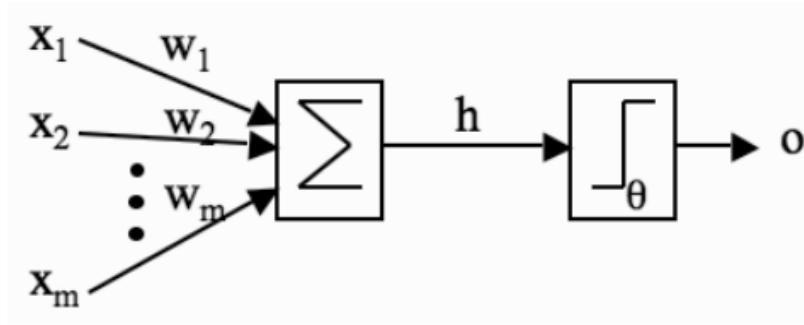


Figure 1.1: McCulloch and Pitts' Neuron.

This model works by first multiplying the input vector x_1, x_2, \dots, x_m by the weight vector w_1, w_2, \dots, w_m . The neuron sums the resulting values and passes that through an activation function. If the sum is greater than the threshold θ then the neuron fires; otherwise it does not. The activation function used in this project is defined in Equation 1.1.

$$o = g(h) = \begin{cases} 1 & \text{if } h > \theta \\ 0 & \text{if } h \leq \theta \end{cases} \quad (1.1)$$

The network consists of a set of input nodes tied to McCulloch and Pitts neurons with weighted connections. This architecture is defined in Figure 1.2 below. The grey nodes are input features or dimensions. The number of features is normally determined by the dataset and pre-processing steps taken. The black nodes are neurons. The number of neurons is arbitrary and doesn't have to match the number of input nodes. Even though it's not shown in Figure 1.2, each neuron still consists of an adder and thresholding function.

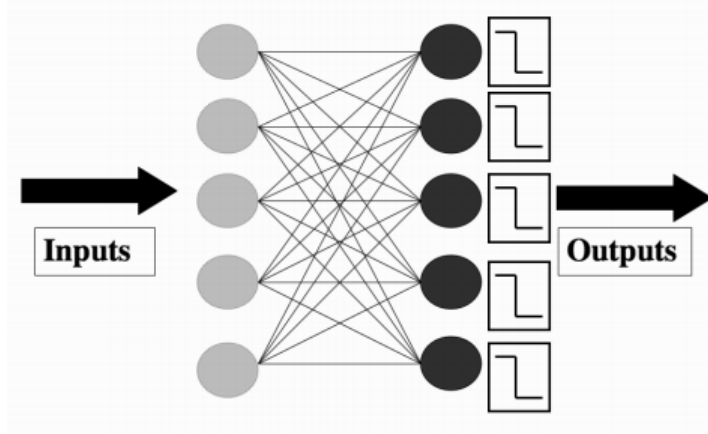


Figure 1.2: The Perceptron network.

The Perceptron is trained iteratively and the weights are updated during each iteration depending on the activations. The weights are updated according to Equation 1.2. Eta represents the learning rate constant which dictates how much the weights change at each iteration. A larger value means the weights will step farther, but risk overshooting the target and never converging. A value that is too small risks a longer training time and never converging if the number of epochs is not sufficient.

$$w_{ij} \leftarrow w_{ij} - \eta(y_j - t_j) \times x_i \quad (1.2)$$

This project will examine the effects of increasing the input dimensionality as well as the influence of the learning rate on the training phase.

Datasets

The two datasets used for this project are comprised of binary matrix representations of the characters *L* and *I*. These datasets were generated by finding all possible unique representations according to a few rules. These rules are given in Table 2.1. Because the generated data should contain as many unique samples as possible, the two datasets are of different sizes. Before presenting the data to the network, each sample is flattened in row-major order.

Rule	Description
1	Must fit within matrix dimensions $a \times a$
2	For each character, height must be between 2 and 3 pixels
3	For each L , width must be between 2 and 3 pixels
4	Generated sample must be unique

Table 2.1: Dataset Generation Rules.

Dataset One

The first dataset of L s and I s contains samples of size 3×3 . Given the constraints in Table 2.1, 18 unique samples were generated. This dataset is perfectly balanced with 9 L samples and 9 I samples. A few select samples are shown in Figures 2.1 and 2.2.

1	0	0
1	0	0
1	0	0

0	1	0
0	1	0
0	1	0

0	0	1
0	0	1
0	0	1

Figure 2.1: I input samples (3×3 matrix).

1	0	0
1	0	0
1	1	0

0	1	0
0	1	0
0	1	1

1	0	0
1	0	0
1	1	1

Figure 2.2: L input samples (3×3 matrix).

Dataset Two

The second dataset of L s and I s contains samples of size 5×5 . Given the constraints in Table 2.1, 83 unique samples were generated. This dataset is slightly imbalanced with 48 L samples and 35 I samples. A few select samples are shown in Figures 2.3 and 2.4.

0	0	0	0	0
0	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0

0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	0

Figure 2.3: I input samples (5×5 matrix).

1	0	0	0	0
1	0	0	0	0
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	1	0

0	0	0	0	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	1
0	0	0	0	0

Figure 2.4: L input samples (5×5 matrix).

Experimentation

Two experiments were performed for this project. The first is meant to observe the influence of the hyper-parameters on the model accuracy, specifically the learning rate and the number of epochs. This experiment was set up by first nesting three for loops. The outer-loop iterates through different values of eta in the range of $1/1000$ and $105/1000$, stepping by $5/1000$. The inner-loop iterates through different values of epochs ranging from 1 to 55 and stepping by 5 epochs. Finally, the inner-inner-loop evaluates the model with the current hyper-parameters using leave one out cross-validation. LOO was chosen because the dataset is relatively small. For each combination of hyper-parameters, the accuracy is calculated. The combination with the best accuracy is logged for analysis. This experiment was performed twice, once per dataset.

The purpose of the second experiment is to observe the impact the learning rate has on model training. This is accomplished by plotting the current epoch or iteration against the misclassification rate. The misclassification rate is calculated according to Equation 3.1.

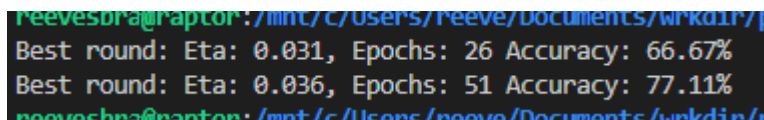
$$\text{Misclassification rate} = \frac{\text{false positives} + \text{false negatives}}{\text{total samples}} \quad (3.1)$$

Because training is the primary concern here, all samples were used. Ten total plots were generated, 5 per dataset, each using a different learning rate. Because the typical value for the learning rate is between 0.01 and 0.1, the learning rates used for this experiment included 0.01, 0.025, 0.05, 0.075, and 0.1.

Analysis

Experiment One

As mentioned previously, the first experiment evaluates the impact that hyper-parameters have on the Perceptron. Also, it compares the accuracy of the model for 3×3 samples against a model using only 5×5 samples. The best results for each dataset are given in Figure 4.1.



```
reevesbra@raptor: /mnt/c:/Users/reeve/Documents/workdir/p
Best round: Eta: 0.031, Epochs: 26 Accuracy: 66.67%
Best round: Eta: 0.036, Epochs: 51 Accuracy: 77.11%
reevesbra@raptor: /mnt/c:/Users/reeve/Documents/workdir/p
```

Figure 4.1: Hyper-parameters with the best accuracy.

The first line in the figure shows the results of the model using 3×3 samples and the second line shows the results of the model using 5×5 samples. Each of these accuracies was computed using leave one out cross-validation due to the small sample sizes. As you can see, the model utilizing 5×5 samples performed better. This means that it was better at generalizing the weights than the model using 3×3 samples. This is because the additional dimensions increased the room for linear separability.

For each model, the best eta value was approximately 0.035. The more interesting result is that the best model using 3×3 samples was trained after only 26 epochs while the best model using 5×5 samples took nearly twice as many epochs to train. Because each data sample in the 5×5 set contained more dimensions, it took longer to find a local minimum.

Experiment Two

The purpose of the second experiment was to observe the impact of the learning rate on the training phase of the Perceptron. The results of this experiment are given in Table 4.1. The column labels signal the value used for eta and the row labels denote the training set sample dimensions. For each plot, the misclassification rate is on the y-axis and ranges from 0.0 to 0.6 and the iteration number is along the x-axis and ranges from 0 to 100. Although not shown here, up to 1000 epochs was tested, but it didn't make much of a difference in terms of the results.

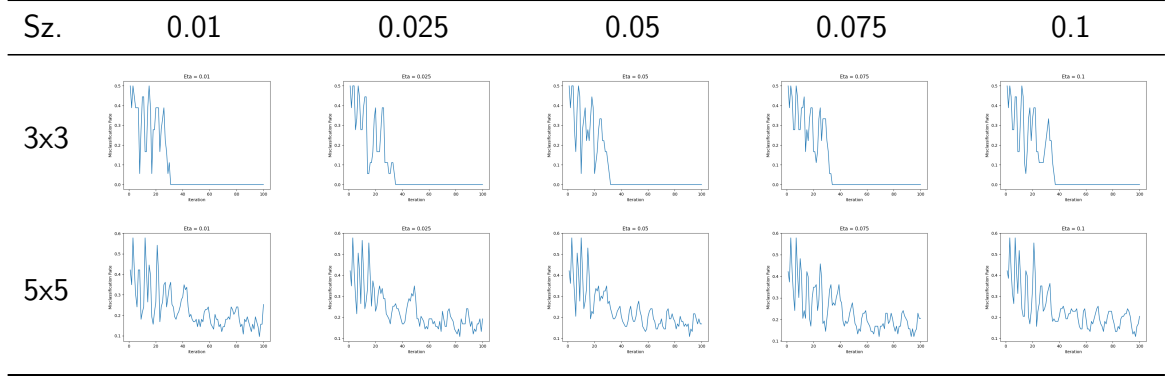


Table 4.1: Effect of learning rate on training.

Although the plots are rather small, we can still observe several characteristics. First of all, the model using the dataset made up of 3×3 samples always seems to converge within 40 or so epochs. As observed with the accuracy results, this doesn't always mean that the model will perform better on test data. The model may be over-fitted which gives poor generalization. With the model using 5×5 samples, the training never converges. Even so, it still performs well with test data. This is likely happening because the samples in the dataset are not linearly separable.

In addition to observing convergence, we can see how the learning rate impacts training. Models evaluated with a smaller eta value appeared to find the local minimum in fewer iterations. This was quite surprising given that larger eta values take larger steps. This likely happened as a result of the models with

larger eta values overshooting and undershooting the appropriate weights.

Conclusion

The Perceptron works surprisingly well as a machine learning model, but it does have its drawbacks. For it to be useful, the dataset should be linearly separable for the most part. It was interesting to see the impact that hyperparameters have on this algorithm. Also, it was interesting to see that training convergence doesn't necessarily create a better model. Even though the model using 3×3 samples converged, the model using 5×5 samples performed better when testing. Overall, this was a fun experiment and we are excited to see how this algorithm can be extended and improved by using multiple layers.

References

- [1] S. Marsland, *Machine Learning An Algorithmic Perspective, Second Edition*. CRC Press, 2015.